Configuration Management at The Edge Service

Thesis

Presented in Partial Fulfillment of the Requirements for the Degree B.S. in Computer Science and Engineering with Honors Research Distinction in the College of Engineering at Ohio State University

By

Yue Zhang,

Department of Computer Science and Engineering

The Ohio State University

2020

Thesis Committee:

Christopher Stewart, Advisor

Thomas Bihari

Abstract

Internet services are increasingly pushed from the remote cloud to the edge sites close to data sources to offer fast response time and low energy footprint. However, software deployed at edge sites must be updated frequently. Performing updates as soon as they are available consumes a large amount of energy. Configuration management tools that install software updates and manage allowed staleness can inflate energy demands, especially when updates interrupt idle periods at the edge site and block processors from entering power-saving modes. Our research studies configuration management policies, their effect on energy footprint and strategies to optimize them. We have observed that policies yielding low energy footprint differ from site to site and over time. We propose a data-driven approach that uses data collected at each edge site to predict an energy-efficient policy and also guards against worst-case performance if data-driven predictions error occurs. We use a novel random-walk approach to manage data-driven policies that yield a low footprint for a representative trace of updates observed at an edge site. We are setting up 4 edge service benchmarks powered by AI inference to create realistic software update traces. Data-driven management achieves energy footprint 1.2-2X lower than the default context-adaptive scheduler (for example the first come first serve scheduler) and lower than approaches inspired by recent research and a little bit higher than the offline optimal.

Dedicated to the students at The Ohio State University

Acknowledgments

The research presented in this thesis was conducted in the ReRout Lab at The Ohio State University under the supervision of Professor Christopher Stewart. The ideas and concepts studied here build upon years of research by prior lab members. I have cited many of those papers in the bibliographic references to acknowledge their impact. I also thank Dr. Thomas Bihari for joining my thesis committee.

Vita

2017-2020 .... B.S. Computer Science and Engineering,
The Ohio State University

Publications

Research Publications

Yue Zhang, Christopher Stewart "Poster: Configuration Management for Internet Services at the Edge: A Data-Driven Approach". *The Fifth ACM/IEEE Symposium on Edge Computing (SEC)*, Nov, 2020.

Fields of Study

Major Field: Department of Computer Science and Engineering

Table of Contents

	Page
Abstract	ii
Dedication	iii
Acknowledgments	iv
Vita	v
List of Figures	vii
1. Introduction	1
2. Design	5
3. System Design	7
4. Results	9
5. Future Work	11
6. Related Work	12
Bibliography	16

List of Figures

Figure **Page**

2.1 AI-Driven IoT 6

3.1 Trace Driven Simulation 7

4.1 Energy Consumption Comparison 10

Chapter 1: Introduction

Internet services are increasingly deployed at edge sites to provide low-latency responses thereby reducing energy footprint [23]. An edge site is a low-power device, with processing capability that allows multiple Internet services deployed at the proximity of the edge site. The low energy footprint of edge sites makes them ideal solutions to deploy different workloads that use machine learning (ML)[40]. The applications have ML models packaged as software and deployed on the IoT device connected to an edge device. Examples include smart cameras, smartwatches, smart traffic controls, drones, and so on [21, 44, 30, 14, 3, 16, 4, 45, 5]. These devices sense surroundings and provide local inference or relay requests to edge sites running internet service.

The software deployed for such internet services must be updated regularly in order to avoid data drift [27] and to improve the performance of internet service. Failure to update the software would result in poor internet service which provides inaccurate and inefficient inferences. When software deployed are allowed to stale, it becomes a challenging problem to perform software updates by consuming minimal energy. Edge sites can enter power saving mode if they decide to skip updating.

Aggressive updating policies that perform the updates as soon as they become available to avoid staleness violation consume enormous energy. The most conservative approach of delaying updates could trap the device from entering deep power-saving mode when many updates arrive that lead to continuous staleness violation. With multiple software updates, deciding whether to skip updates or how many updates to perform and in what order to perform updates would be a challenging problem to tackle. Given the trace in advance, the complexity of an offline solution to such a problem is NP-Hard [2].

Given a trace of updates such that factors do not change, there exist techniques such as random walks [2] or reinforcement learning[1] in order to find the best scheduling policy to perform software updates in an energy-efficient manner. Continuing with fixed factors for a period of time, such a scheduling policy performs updates consuming minimal energy.

In reality, there are multiple factors that affect a policy performing software updates such as update rate, allowed staleness, number of applications deployed, idle time length, bandwidth, and size of the update[7]. Given a historic trace of updates with fixed factors, one could find an energy-efficient updating policy offline and apply the policy to updates that arrive in the future. Changes to these factors worsen the policy, forcing it to consume excess energy. The policy needs to be updated in order to be energy-efficient. We do not know these factors in advance and subtle changes to these factors affect the currently preferred software scheduling policy. Over time, when such factors change and worsen the energy consumption, there should be changed to the current update policy or fall back to a safe updating policy that can sustain the demand and factors better than the current update policy. Such policies differ from site to site and each edge site requires a configuration management tool to perform energy-efficient software updates.

Our research proposes data-driven configuration management for edge sites. Configuration management, inspired solely by data collected from edge sites, predicts an effective policy to perform software updates. Configuration management not only encourages multiple policies that perform energy-efficient updates but guards against the worst-case when such policies worsen due to change in factors and data-driven predictions error[22]. Configuration management periodically collects traces and performs random walk to select energy-efficient software updating policy. We build configuration management and implement a data-driven approach to perform software updates.

We test data-driven configuration management using the simulation of realistic internet services for different workloads and edge devices. The workloads include image detection on Mnist(MNIST), intruder detection(Intrude), traffic sign recognition(TSR), and human activity recognition(HAR). The edge site could be HP-Laptop, raspberry-pi with the provision of adding different accelerators on top of them for faster inference. We compare our data-driven approach to state-of-the-art inspired systems such as GAIA[15], SCEDA[1], which proposed a model synchronization mechanism that ensures staleness by reinforcement learning. Data-Driven configuration management achieves energy footprint 1.2-2X lower than first come first serve (FCFS) scheduling for all workloads, and a little bit higher than the offline optimal.

The remainder of the paper is organized as follows: Chapter 2 describes the design of configuration management. Chapter 3 describes the system design of configuration management. Chapter 4 presents results, Chapter 5 presents future work, and Chapter 6 presents related work.

Chapter 2: Design

AI-Driven IoT encompasses AI applications that are deployed on edge computing platforms for fast inference and low-latency results. The Applications have ML models packaged as software and deployed on the IoT device connected to an edge device. Multiple such edge devices could serve numerous IoT devices with AI processing capability. Figure 2.1 depicts the AI-Driven IoT workflow. ML models are trained on cloud servers with sufficient compute capability and pushed to edge devices for updates. Edge devices perform these updates according to the scheduler deployed.

[2] suggests that given the option of performing updates or putting edge device to sleep in an inter-arrival period, the update order impacts the amount of energy incurred for processing updates. However, given the option to choose between static (context-aware) scheduler and a dynamic scheduler (which can adapt to context changes), we need to strike a balance of picking scheduler over period of time. We do not know when context changes and reverts back, number of updates, inter-arrival rate in advance. This research focuses on implementing an online scheduler which picks correct scheduler in order to minimize energy cost to perform application updates.

In order to solve the problem in hand, we might think of deploying a static context-aware scheduler or dynamic adaptive scheduler for entirety. However, deploying a

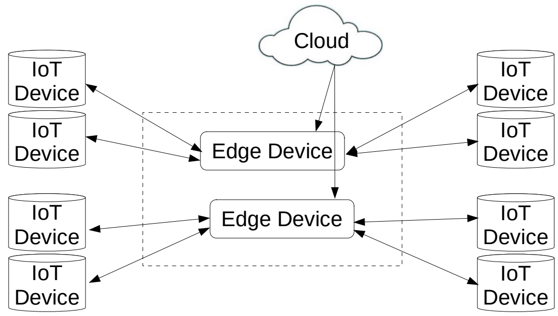

Figure 2.1: AI-Driven IoT

static context aware scheduler or adaptive scheduler for entirety would be a costly operation. It consumes orders of magnitude of additional energy as context changes happen often in practice.[20] We must devise an online algorithm which has good competitive guarantees and perform well in practice. Here, we are inspired by prior work using online algorithms in systems management [41, 42, 43].

Chapter 3: System Design

In this section we highlight how system is designed and integration of this contribution to a working AI-Driven IoT System. Figure 3.1 shows the trace driven simulation. Powerful Google Cloud instances are used for retraining the ML model, inference data updating and these application update tasks are pushed from cloud to edge. For retraining, we use some of the retraining policies from continuum such as best-effort, periodic and cost-aware retraining policies[39]. Best-effort triggers restraining as soon as new data is available, periodic triggers retraining periodically and cost-aware triggers retraining such that energy to update will be minimized.

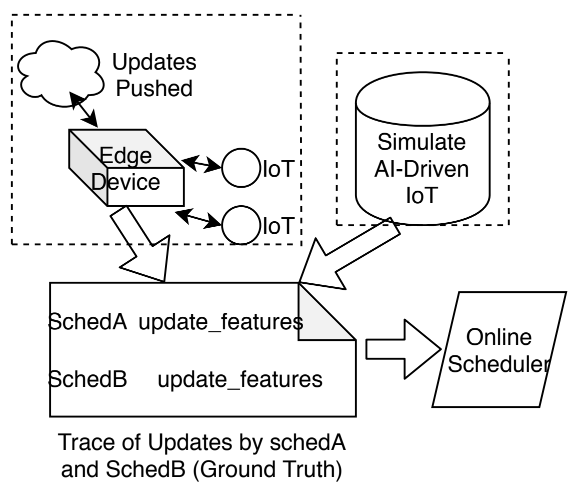

Figure 3.1: Trace Driven Simulation

Edge devices are HP laptop with 2.3 GHz i3 processor, 4 GB RAM and Ubuntu Operating system; Raspberry Pi 4 with 1Gb RAM and Raspian operating system. We have also added accelerators such as Google Coral USB, Nvidia Jetson Nano devkit to our edge devices to provide fast inferences. Docker is installed on every edge device and the application running on edge device is containerized. Every task that is pushed would be an updated Docker image to be deployed on the edge device.

To deploy our contribution, we design two schedulers: 1) context-aware scheduler which has learned the context from past traces and 2) context adaptive scheduler which performs updates in a first come first server (FCFS) manner. There should be context configuration scripts which specifies a time-stamped context changes over period of time. The updates are pushed and are held at the edge device in a queue. The edge device processes inference and during idle time it processes the updates or puts the device to deep sleep. The scheduler deployed at the edge is very crucial in order to minimize the energy required to process the updates. The online scheduler switches schedulers for every inter-arrival period by deciding to pick context-aware versus choosing context adaptive scheduler.

With this deployment and running a trace, one could get energy to process the updates using offline optimal, online scheduler and default context-adaptive scheduler.

Chapter 4: Results

We use a random-walk approach for scheduling updates and compare the energy consumption with the first come first serve(FCFS) scheduling. Both have been used on 4 edge service benchmarks powered by 50 AI inference for each. AI inferences that perform different update rates have been used to simulate realistic traces. Each edge service benchmark with different convolutional neural network architecture has been used to simulate different software updates. The FCFS scheduling performs updates whenever the update is available without staleness. Our random-walk approach randomly chooses to install the update or delay within staleness limits and we calculate the average energy consumption of several random-walk choices. The staleness of the random-walk approach has been limited within 3 versions. Both approaches achieve the same update version at the end of the experiment with different energy consumption. Figure 4.1 shows the energy consumption level of FCFS scheduling versus the random-walk approach. The average energy consumption of the random-walk approach consumes energy footprint 1.2-2X lower than the FCFS scheduling for all 4 benchmarks.

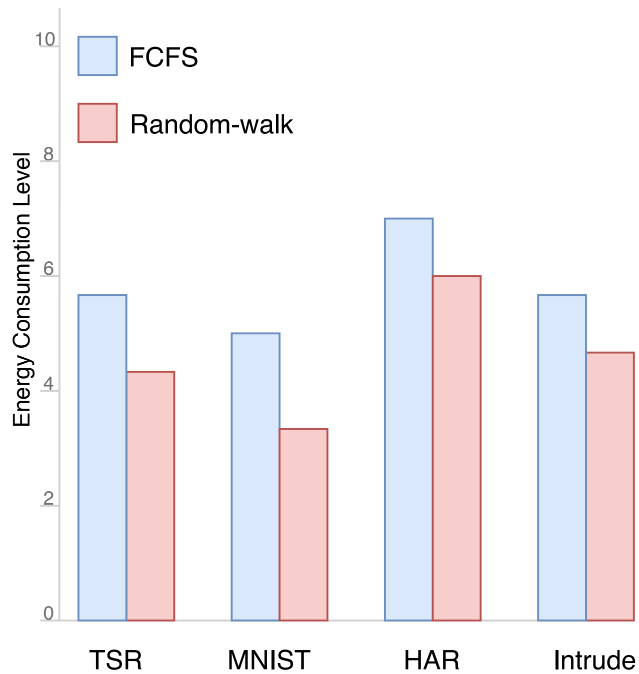

Figure 4.1: Energy Consumption Comparison

Chapter 5: Future Work

We plan to continue to improve our data-driven configuration management tool by implementing a machine learning predictor to pick up between context-aware scheduler and dynamic adaptive scheduler. Given that contexts can change over a period of time, the update rate is skewed and context changes can revert back, our goal is to minimize the energy incurred to process updates. In order to do this, for every inter-arrival period a scheduler that consumes the lowest update cost should be chosen. However, in reality, we do not know the workload demand and the context factors in advance. Thus, we plan to implement a machine learning predictor by random forest approach to make the choice between context-aware scheduler and dynamic adaptive scheduler at each inter-arrival period based on the past scheduling choices, update rates, total energy consumption, and staleness tradeoff. The error rate of machine learning predictors would be considered under different scheduling scenarios. We would prove that our data-driven management approach would achieve lower energy footprint than the default context-adaptive scheduler with the error rate in considering. We would also compare our result with GAIA[15] and SCEDA[1], which also proposed a synchronization mechanism to ensure the staleness as well as model correctness.

Chapter 6: Related Work

Naveen et al proposed Energy, Latency, and staleness tradeoffs for AI-Driven IoT [2]. The researchers re-purposed progressing sampling using energy, storage, and latency metrics from [24]. With simulations, they showed that traditional scheduling techniques like FCFS, SJF, and LRU aren't energy-efficient and there is room for better schedulers. The Random walk results show a significant gap between 75th, 95th percentile, and high ranking policies. The design of experiments shows that Energy, latency, the architecture of IoT devices, and staleness are some important factors to be considered in designing a scheduler and there is room to design an online scheduler to handle machine learning updates [2].

Peak aware energy scheduling (PAES) problem involves choosing between the electricity grid and local generator to satisfy the demands that arrive over a period of time (usually a month) [22]. A peak charge to serve peak demand is usually 100-100x costlier than a charge from the grid. This is an interesting online scheduling problem, given a grid and local generator, how do you schedule the demand such that you can minimize the electricity cost. The past work initially uses a local generator and switches to a grid when the break-even point is reached. This work proposed deterministic and randomized online algorithms which provide bounds on robustness and consistency as the competitive ratio of the online algorithm. The recent work proposed learning-assistance as an extension to the PAES problem. Given machine learning prediction guarantees, researchers have tried to improve the robustness and consistency of existing deterministic and randomized algorithm [22].

Huanshi et al proposed Continuum: A platform for cost-aware, low-latency continual learning [39]. Machine learning applications deployed in a dynamic environment would require model updates as data shifts and environment changes. The researchers have introduced different retraining policies to facilitate continual learning. Periodic, best-effort, uniform, and cost-aware were some of the retraining policies introduced. Cost-aware retraining, which takes the retraining cost into consideration, is proved to be 2-competitive. Continuum is open-sourced and ideas from paper can be used to replicate retraining policies while deploying machine learning applications [39].

Edge data analytics bring processing power in the proximity of data sources, reduce the network delay for data transmission, allow large-scale distributed training, and consequently help meeting real-time requirements [1]. However, due to data diversity, machine learning models suffer from the inconsistent state from staleness. This work proposed a model synchronization mechanism for distributed and stateful data analytics considering the insularity, concept drift, and connectivity issues in edge data analytics with the balance of model performance and timeliness [1]. By using online reinforcement learning, the researchers proposed a low computational and automatic way to update without additional data monitoring.

Running machine learning (ML) algorithms over wide-area networks (WANs) with the limited WAN bandwidth can be extremely slow due to the large-scale data and frequent communication over WANs. This work introduced a new, general geo-distributed ML system, Gaia, that decouples the communication within a data center [15]. A new ML synchronization model, Approximate Synchronous Parallel (ASP), has been used to dynamically eliminate insignificant communication between data centers while guaranteeing the correctness of ML algorithms [15].

Acknowledgments: The research presented in this thesis was conducted in the ReRout Lab at The Ohio State University under the supervision of Professor Christopher Stewart. The ideas and concepts studied here build upon years of research by prior lab members. I have cited many of those papers in the bibliographic references to acknowledge their impact.

Bibliography

[1] A. Aral, M. Erol-Kantarci, and I. Brandić. Staleness control for edge data analytics. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 4(2):1–24, 2020.

[2] N. T. Babu and C. Stewart. Energy, latency and staleness tradeoffs in ai-driven iot. In *Symposium on Edge Computing*, 2019.

[3] S. Bhattacharya and N. D. Lane. From smart to deep: Robust activity recognition on smartwatches using deep learning. In *Pervasive Computing and Communication Workshops*, 2016.

[4] J. Boubin, N. Babu, C. Stewart, J. Chumley, and S. Zhang. Managing edge resources for fully autonomous aerial systems. In *ACM Symposium on Edge Computing*, 2019.

[5] J. Boubin, J. Chumley, C. Stewart, and S. Khanal. Autonomic computing challenges in fully autonomous precision agriculture. In *IEEE International Conference on Autonomic Computing*, 2019.

[6] A. Chakrabarti, C. Stewart, and S. Parthasarathy. Green- and heterogeneity-aware partitioning for data analytics. In *IEEE International Workshop on Green and Sustainable Networking and Computing*, 2016.

[7] J. Chen and X. Ran. Deep learning with edge computing: A review. *Proceedings of the IEEE*, 107(8):1655–1674, 2019.

[8] D. Chiu, C. Stewart, and B. McManus. Electric grid balancing through low-cost workload migration. In *ACM GreenMetrics Workshop*, 2012.

[9] N. Deng. Carbon-aware cloud application. In *IEEE Network Operations and Management Symposium*, 2012.

[10] N. Deng, C. Stewart, J. Kelley, D. Gmach, and M. Arlitt. Adaptive green hosting. In *IEEE International Conference on Autonomic Computing*, 2012.

[11] N. Deng, C. Stewart, and J. Li. Concentrating renewable energy in grid-tied datacenters ¡br¿(awarded of third best paper). In *IEEE International Symposium on Sustainable Systems Technology*, 2011.

[12] P. Derosa, K. Shen, C. Stewart, and J. Pearson. Realism and simplicity: Disk simulation for instructional os performance evaluation. In *ACM SIGCSE Technical Symposium*, 2006.

[13] J. Ding, R. Cao, I. Saravanan, N. Morris, and C. Stewart. Characterizing service level objectives for cloud services: Realities and myths. In *IEEE International Conference on Autonomic Computing*, 2019.

[14] S. Flutura, A. Seiderer, I. Aslan, C. T. Dang, R. Schwarz, D. Schiller, and E. Andre. Drinkwatch: A mobile wellbeing application based on interactive and cooperative machine learning. In *International Conference on Digital Health*, 2018.

[15] K. Hsieh, A. Harlap, N. Vijaykumar, D. Konomis, G. R. Ganger, P. B. Gibbons, and O. Mutlu. Gaia: Geo-distributed machine learning approaching LAN speeds. In *Symposium on Networked Systems Design and Implementation*, 2017.

[16] M. A. Kader, E. Bastug, M. Bennis, E. Zeydan, A. Karatepe, A. S. Er, and M. Debbah. Leveraging big data analytics for cache-enabled wireless networks. In *IEEE Globecom Workshops (GC Wkshps)*, 2015.

[17] J. Kelley, C. Stewart, N. Morris, D. Tiwari, Y. He, and S. Elnikety. Obtaining and managing answer quality for online data-intensive services. In *ACM Transactions on Modeling and Performance Evaluation of Computing Systems*, 2017.

[18] J. Kelley, C. Stewart, D. Tiwari, and S. Gupta. Adaptive power profiling for many-core hpc architectures. In *IEEE International Conference on Autonomic Computing*, 2016.

[19] J. Kelley, C. Stewart, D. Tiwari, Y. He, S. Elnikey, and N. Morris. Measuring and managing answer quality for online data-intensive services. In *IEEE International Conference on Autonomic Computing*, 2015.

[20] M. Kim, J. Lee, Y. Kim, and Y. H. Song. An analysis of energy consumption under various memory mappings for fram-based iot devices. In *2018 IEEE 4th World Forum on Internet of Things (WF-IoT)*, 2018.

[21] N. D. Lane, S. Bhattacharya, A. Mathur, P. Georgiev, C. Forlivesi, and F. Kawsar. Squeezing deep learning into mobile and embedded devices. *IEEE Pervasive Computing*, 16(3):82–88, 2017.

[22] R. Lee, M. H. Hajiesmaili, and J. Li. Learning-assisted competitive algorithms for peak-aware energy scheduling. In *arXiv preprint arXiv:1911.07972*, 2019.

[23] P. Liu, D. Willis, and S. Banerjee. Paradrop: Enabling lightweight multi-tenancy at the network's extreme edge. In *ACM Symposium on Edge Computing*, 2016.

[24] S. Liu, Y. Lin, Z. Zhou, K. Nan, H. Liu, and J. Du. On-demand deep model compression for mobile devices: A usage-driven model selection framework. In *Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services*, pages 389–400, 2018.

[25] N. Morris, S. Renganathan, C. Stewart, R. Birke, and L. Chen. Sprint ability: How well does your software exploit bursts in processing capacity? In *IEEE International Conference on Autonomic Computing*, 2016.

[26] N. Morris, C. Stewart, L. Chen, R. Birke, and J. Kelley:. Model-driven computational sprinting. In *ACM European Conference on Computer Systems*, 2018.

[27] J. Quionero-Candela, M. Sugiyama, A. Schwaighofer, and N. D. Lawrence. Dataset shift in machine learning. MIT Press, 2009.

[28] R. Rao, C. Stewart, A. Perez, and S. Renganathan. Assessing learning behavior and cognitive bias from web logs. In *IEEE Frontiers in Education*, 2018.

[29] S. Renganathan, C. Stewart, A. Perez, R. Rao, and B. Braaten. Preliminary results on an interactive learning tool for early algebra education. In *IEEE Frontiers in Education*, 2017.

[30] F. Shahmohammadi, A. Hosseini, C. E. King, and M. Sarrafzadeh. Smartwatch based activity recognition using active learning. In *International Conference on Connected Health: Applications, Systems and Engineering Technologies*, 2017.

[31] C. Stewart, A. Chakrabarti, and R. Griffith. Zoolander: Efficiently meeting very strict, low-latency slos. In *IEEE International Conference on Autonomic Computing*, 2013.

[32] C. Stewart, T. Kelly, and A. Zhang. Exploiting nonstationarity for performance prediction. In *European Conference on Computer Systems*, 2007.

[33] C. Stewart, M. Leventi, and K. Shen. Empirical examination of a collaborative web application. In *IEEE International Symposium on Workload Characterization (special session on Benchmark Innovation)*, 2008.

[34] C. Stewart and K. Shen. Performance modeling and system management for multi-component online services. In *Symposium on Networked Systems Design and Implementation*, 2005.

[35] C. Stewart and K. Shen. Some joules are more precious than others: Managing renewable energy in the datacenter. In *ACM Workshop on Power Aware Computing and Systems*, 2009.

[36] C. Stewart, K. Shen, S. Dwarkadas, M. L. Scott, and J. Yin. Profile-driven component placement for cluster-based online services. In *International Middleware Conference*, 2004.

[37] C. Stewart, K. Shen, A. Iyengar, and J. Yin. Entomomodel: Understanding and avoiding performance anomaly manifestations (winner of best paper award). In *IEEE Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, 2010.

[38] C. Stewart, M. Zhong, K. Shen, and T. O'Neill. Comprehensive depiction of configuration-dependent performance anomalies in distributed server systems. In *USENIX Hot Topics in System Dependability*, 2006.

[39] H. Tian, M. Yu, and W. Wang. Continuum: A platform for cost-aware, low-latency continual learning. In *Proceedings of the ACM Symposium on Cloud Computing*, pages 26–40, 2018.

[40] X. Wang, Y. Han, V. C. Leung, D. Niyato, X. Yan, and X. Chen. Convergence of edge computing and deep learning: A comprehensive survey. In *IEEE Communications Surveys & Tutorials*, 2020.

[41] Z. Xu, N. Deng, C. Stewart, and X. Wang. Cadre: Carbon-aware data replication for geo-diverse services. In *IEEE International Conference on Autonomic Computing*, 2015.

[42] Z. Xu, C. Stewart, N. Deng, and X. Wang. Blending on-demand and spot instances to lower costs for in-memory storage (winner best-in-session presentation). In *IEEE International Conference on Computer Communications*, 2016.

[43] Z. Xu, C. Stewart, and J. Huang. Elastic, geo-distributed raft. In *ACM International Symposium on Quality of Service*, 2019.

[44] S. Yi, Z. Hao, Z. Qin, and Q. Li. Fog computing: Platform and applications. In *Hot Topics in Web Systems and Technologies (HotWeb)*, 2015.

[45] S. Zhang and C. Stewart. Computational thinking curriculum for unmanned aerial systems. In *IEEE National Aerospace and Electronics Conference*, 2019.